Video based Animation Synthesis with the Essential Graph

Adnane Boukhayma, Edmond Boyer

Video based Animation Synthesis with the Essential Graph

Adnane Boukhayma, Edmond Boyer
INRIA Rhône-Alpes
Grenoble
{adnane.boukhayma,edmond.boyer}@inria.fr

Abstract

We propose a method to generate animations using video-based mesh sequences of elementary movements of a shape. New motions that satisfy high-level user-specified constraints are built by recombining and interpolating the frames in the observed mesh sequences. The interest of video based meshes is to provide real full shape information and to enable therefore realistic shape animations. A resulting issue lies, however, in the difficulty to combine and interpolate human poses without a parametric pose model, as with skeleton based animations. To address this issue, our method brings two innovations that contribute at different levels: Locally between two motion sequences, we introduce a new approach to generate realistic transitions using dynamic time warping; More globally, over a set of motion sequences, we propose the essential graph as an efficient structure to encode the most realistic transitions between all pairs of input shape poses. Graph search in the essential graph allows then to generate realistic motions that are optimal with respect to various user-defined constraints. We present both quantitative and qualitative results on various 3D video datasets. They show that our approach compares favourably with previous strategies in this field that use the motion graph.

1. Introduction

3D animation has become a crucial part of digital media production with numerous applications, in particular in the game and motion picture industry. Traditionally, virtual characters are created and animated either manually with, for instance user defined keyframes, or automatically with motion capture data. While manual animations are usually tedious and require artistic expertise, motion capture data allows to animate characters with real motion information extracted from real performances. In addition to significantly reduce the animation cost, the benefit of motion capture data also lies in the fact that movements are real and, therefore, ensure realism to the created animations. Going a step further, recent seminal works in computer animation [6] propose to bypass purely virtual characters and to use real videos for both the creation and the animation of characters. The advantage of this strategy is twofold: it reduces the creation cost and increases realism by considering only real data. Furthermore, it allows to create new motions, for real characters, by recombining recorded elementary movements. In this paper, we consider this synthesis of real motions using videos and propose a new solution to automatically build animations that are as real as possible.

Multi-view capture systems, *e.g.* [26], can produce 3D videos of human movements. In these 3D videos, shape information is generally encoded in the form of temporal mesh sequences, coherent or not. In order to recombine elementary movements, motion information could be parametrized using skeletal articulated models, as with traditional motion capture systems, and concatenated based on such parametrization. While methods exist that provide articulated motion information, *e.g.* [29, 7], this strategy suffers anyway from two drawbacks: First the identification of such information is difficult to perform robustly with generic surfaces that do not always evolve according to the assumed articulated motion model, as with clothes for instance. Second, the use of an intermediate skeletal model makes it difficult to guarantee realism for transitions that finally concern surfaces around skeletons. Another strategy is to avoid the intermediate motion model and to directly consider surfaces or meshes. This is the direction taken by [10] who proposed to extend motion graphs, originally designed for skeletal motion capture data [15], to structure and recombine motion information with mesh sequences. In such a graph, edges connect consecutive meshes within a sequence and new edges are added between pair of sequences and with respect to a similarity cost between meshes. Although simple and intuitive, motion graphs do not yet guarantee global optimality of created motion paths since such paths are composed of locally optimal edges, *i.e.* between two sequences, where other paths, through edges not in the motion graph, with better global costs can exist. In order

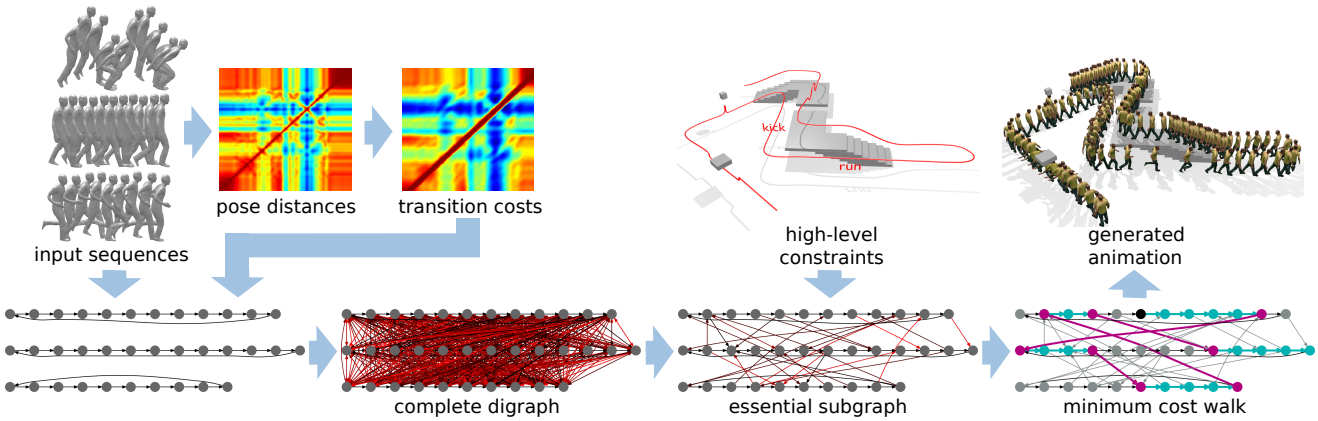

Figure 1. Overview of the proposed video based animation synthesis pipeline.

to find such paths and to optimally exploit all the available shape poses, a fully connected graph must be considered.

In light of this, we consider animations that are globally optimal with respect to a defined realism criterion. This criterion derives from physical considerations and takes into account a surface interpolation cost, that measures the amount of surface deformation along a motion segment, as well as a duration cost that accounts for the number of frames of a segment. The first cost tries to preserve shape consistency along a motion segment while the second tries to minimize the number of poses within a segment. Our approach first builds a fully connected graph where every shape pose is connected to all others and where edges weight direct transitions between poses with respect to the realism criterion. To estimate the edge weights, each transition is optimized through dynamic time warping and variable blended segments lengths. In a second step, the graph is pruned and the essential subgraph, that is the union of all shortest path trees rooted at every node, is extracted. The essential subgraph is an optimal structure that contains only the necessary edges to generate the most realistic graph transitions between all pairs of input frames. Finally, the resulting structure is used to generate motion that satisfies user-specified spatial, temporal and behavioural constraints as demonstrated in our results, by solving a standard search optimization problem over the essential graph and with the appropriate cost function. At a lower level, our approach combines mesh non-linear interpolation, with 2D rigid alignments to generate the spatially continuous motion sequence corresponding to a graph walk in the essential subgraph.

To summarize, our contribution is twofold:

- We introduce a transition approach that combines dynamic time warping and variable length blended segments to generate mesh based interpolated transitions that are optimal with respect to a realism criterion.
- We propose the essential graph as an optimal structure to organize several pose sequences of a person movements and from which minimal cost paths are easily identified between two given poses.

In order to evaluate the benefit of the essential graph over the motion graph, we compare both graph structures in a similar context, quantitatively with our criterion, and qualitatively with visual results (see also the accompanying video).

The remainder of the paper is as follows. Section 2 reviews related work. In section 3, we present our graph representation and we explain motion synthesis in Section 5. Section 6 compares our approach with a standard motion graph based strategy and shows results before concluding in Section 7.

2. Related work

2.1. Motion Data Structures

Traditional motion capture systems provide sparse information on moving shapes in the form of marker positions. Using such markers positioned at joint locations of, for example a human body, it is possible to track the poses of an associated skeletal model. A large body of work is considering such input data to animate virtual characters. To this aim, motion graphs [15] are largely used to organize and structure recorded movement sequences of a shape. A motion graph is composed of nodes that represent shape poses and of edges that represent transitions between them. To build such a graph, every consecutive nodes in a temporal sequence are connected. Edges are added to this graph by considering a pose and velocity similarity matrix that encode all possible pairs of poses between two sequences. Local minima found in this matrix, with sometimes thresholding to reduce the number of resulting transitions, define the new edges in the motion graph.

Following motion graphs, several solutions have been

proposed in the literature. [17] builds first a directed graph in a similar manner to a motion graph, then subsequently builds a statistical model over it for a more intuitive control. [1] also creates a graph where nodes represent sequences, and transition edges are added between sequence poses if a similarity cost between them is lower than a user specified threshold. A hierarchical randomized search is used then to generate motions. [22] introduces a structure called interpolated motion graphs that consists of interpolating time-scaled paths within a standard motion graph. [33] increases the connectivity in a motion graph by using multi-target blending to generate intermediate interpolated motions between some of the dataset sequences and building a well connected motion graph that incorporates these new motions. [20] constructs an optimization-based graph combining constrained optimization and motion graphs. In the method we present, we also use a graph representation that connects frame poses in temporal sequences. However we consider mesh vertices instead of skeleton joint locations and, more importantly, our graph representation enables paths between any two frames that satisfy global optimality, over pose recombinations, and with respect to surface deformation and transition duration.

Some statistical approaches have been proposed to solve the animation problem for skeletal data. [28] applies dimension reduction and clustering to the input dataset and constructs a Markov chain with these clusters as states in a first level. A second level consists then of a hidden Markov model that relates the states of the Markov chain to the motion examples. Given a source and destination key frames, the corresponding sequence of clusters is found in the first level, and the most likely succession of motion segments is found in the second. More recently, [18] proposes a graph where nodes represent motion primitives and edges transitions between them. Each node stores a statistical model learned from similar motion segments associated to the same primitive and a transition distribution function is associated to every directed edge. However, unlike skeletal motion capture data and besides their very high dimensionality, 3D mesh motion datasets that are currently available represent mostly limited sets of movements. Thus, such statistical strategies do not adapt to this specific type of data. Additionally, and unlike [18], our solution also solves for random acyclic motion, such as an unstructured dancing sequence (see results for *Cathy*), which seems hard to decompose into distinctive simple motion primitives. In a graph-less approach, [11] clusters motion segments and pre-compute a table defining intermediate segments that are used to blend all pairs of cluster medoids. In order to generate a transition from a source segment to a target segment, these two segments are thus blended together with their respective cluster medoid intermediaries. Although this work enables, in principle, transitions between all pairs of frames in the dataset, it requires transition scoring, motion segment clustering and multi-way frame blending that would be difficult to achieve with 3D mesh sequences and without contact information.

2.2. Motion Similarity

Measuring motion similarity is an important aspect in animation synthesis. Although [14] argued that logical motion similarity does not necessarily imply numerical motion similarity, most of the works mentioned previously defined some quantitative pose and velocity similarity metric to estimate motion similarity. For instance, [15] computes a weighted sum of squared distances between point clouds driven by the skeleton and formed over a window of frames that are 2D rigidly aligned, while [17] uses a weighted sum of euclidean differences of joint velocities, skeleton root positions and geodesic distances between joint orientations. Our approach considers mesh based pose distances but is independent of the distance function used for that purpose.

2.3. Motion Transition

Another important component of motion synthesis is the generation of transitions between sequences. Gradual frame blending remains yet a reasonable and intuitive solution that is widely used in the literature, either in a start-end [15], center-aligned [1] or left aligned [17, 22] transition form. However, it might generate unpleasant results if incorrectly applied, especially with linear interpolation and unless costly space-time constraints and inverse kinematics are used as in [21]. These limitations can anyway be substantially reduced with motion dynamic time warping. Following a strategy similar to [13], we use dynamic time warping but extend the framework to transitions between mesh sequences. In addition, inspired by the study of interpolated transitions in [30] that emphasizes the importance of a suitable transition duration, we allow such duration to vary during a transition.

2.4. Mesh Sequences

Only a few works consider yet mesh sequences as input data for motion synthesis. They were first introduced in [10] in association with motion graphs and frame juxtaposition for transitions, with therefore the limitations mentioned before for transitions and motion graphs. [12] also uses motion graphs but over separate groups arranged in a tree-structured kinematic hierarchy forming the global object. Linear interpolation in the local coordinate frame of each group was sufficient for their type of data. In [34], the animation is solved in the skeleton domain using linear blend skinning. Motion Transitions are selected by simple thresholding of a transition cost calculated using the pose parameters and their first derivatives, as in [1]. Our objective is to solve the animation in the mesh domain, remov-

ing therefore the issue of mesh/skeleton conversion, and we propose an elaborated model for the selection and generation of transitions.

More recently, in a work similar to [14] with skeletal data, [5] achieves motion synthesis by temporally aligning and blending logically similar 3D mesh motion sequences, and mapping the blend weights to a high level parameter, thus creating a parametrized motion space. A Parametric motion graph is then built by connecting several parametric motion spaces through pre-computed transitions, as in [9] for skeletal data, or by computing optimal transitions at runtime, as in [6] for 3D meshes. The theory behind parametric motion graphs is similar to that of Fat graphs [23], inspired by the work of [8]. These methods allow motion synthesis with precise control but only for specific input data and require a high level of supervision. Our objective is different and improves over [10] with globally optimal motion sequences generated from a general set of input sequences.

3. Graph model

This section details the steps involved in the construction of the essential graph for mesh based motion segments. The input data is a set of temporally aligned sequences of 3D triangulated meshes, *i.e*. meshes present the same topology and vertex connectivity over all sequences. These sequences represent a model, typically but not necessarily a human, undergoing arbitrary movements with no prior restrictions on the nature of the movements. It should be noticed here that while we consider meshes as input, the graph construction is independent of the pose model used and could be applied with other models including articulated models.

A directed weighted graph structure is drawn from the input sequences, where nodes are sequence frames, corresponding to shape poses, and edges represent possible transitions between them. Edge weights account for both the surface interpolation cost of transitions and their duration. Initially, edges in the graph connect successive frames in the input sequences, as shown in figure 1. Since these edges correspond to natural transition, their surface interpolation cost is low, zero in practice in our implementation, and their duration cost is one time unit (assuming uniform temporal sampling). From this initial graph, a fully connected graph is built where additional edge weights are estimated using a static pose similarity metric and dynamic time warping, as detailed in section 3.2. In a last step, the essential graph is extracted from the fully connected graph by pruning edges that do not contribute to minimal cost paths in the graph.

3.1. Pose similarity metric

We define a static distance function between two given shape poses. Such basic distance will be used further at various stages of our approach, for instance when estimating the cost of a dynamic transition that involves several intermediate poses. Given two poses i and j, we first rigidly align them by finding the rigid transformation that minimizes the weighted sum of squared distances between mesh vertices, where the rigid transformation is composed of a translation in the horizontal plane (*i.e*. the plane parallel to the motion) and of a rotation around the vertical axis. Weights are used, for instance, to give more importance to torso vertices with humans. Once aligned, one could take the residual of the alignment cost as the distance between the poses. However, this residual does not account for the mesh geometry [24] . We use instead local deformations [2] or deformation gradients as referred to in [27, 32], as explained below.

For every pair of corresponding triangles (v_1, v_2, v_3) and (v'_1, v'_2, v'_3) in a pair of meshes, we compute the following unique transform matrix:

$$T = (v_1 - v_3, v_2 - v_3, n)^{-1} \cdot \left(v'_1 - v'_3, v'_2 - v'_3, n'\right), \quad (1)$$

where n and n' are the triangle unit normals. We perform polar decomposition on this linear transformation matrix to extract the orthogonal rotation component R_m, and symmetric scale/shear component S_m for triangle m, and also store them for later purposes such as pose interpolation:

$$T = R_m S_m. \quad (2)$$

Combining the Riemannian metric of the Lie goup of rotations $SO(3)$, and Frobenius norm for symetric 3x3 matrices, we define our pose similarity metric between frames i and j as follows:

$$d(i,j) = \sum_{m \in Triangles} \frac{1}{\sqrt{2}} \| \log(R_m) \|_F + w_m \| S_m - I_3 \|_F. \quad (3)$$

Through weighting factors w_m, we give more importance to the rotation term. In practice, we use the same value $w_m = 0.01$ for all mesh triangles. In order to validate this distance function, we have conducted experiments with skeleton based mesh animations for which skeleton pose distances are assumed to capture exact shape pose distances. Comparisons between distance matrices obtained with pairs of sequences show that the above distance function gives better results than the sum of squared distances mentioned previously, *i.e*. distances (3) present higher correlations with skeleton pose distances.

3.2. Graph edge weights

Weights attributed to edges in our graph should ideally capture the realism of the transitions they represent. Assuming that a realistic transition should involve as little surface deformation as possible, we introduce a metric that measures the minimum surface interpolation cost between any pair of nodes in the graph. It involves several intermediate poses for that purpose, which makes it different from the static pose similarity. This metric, in addition to the

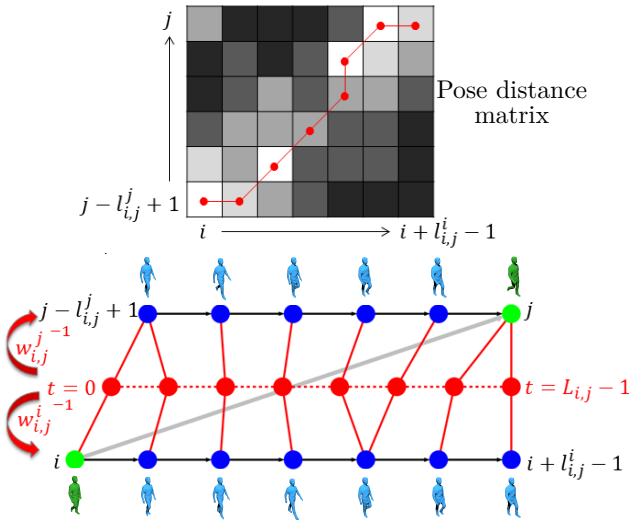

Figure 2. In order to generate an interpolated transition between frame i and j, we gradually blend source and destination segments $[i, i + l^i_{i,j} - 1]$ and $[j - l^j_{i,j} + 1, j]$ using dynamic time warping that finds the minimum-cost path joining element $(i; j - l^j_{i,j} + 1)$ to element $(i + l^i_{i,j} - 1; j)$ in the pose distance matrix between the two segments frames. This path should be continuous, not reverse direction, and not exceed 3 consecutive vertical or horizontal steps. The resulting path defines the temporal warps $w^i_{i,j}$ and $w^j_{i,j}$.

transition duration, allows us to quantitatively evaluate the realism of a transition.

In order to generate a smooth interpolated transition from frame i in a source sequence to a frame j in a destination sequence, we consider a window of successive frames of size l^i in the source sequence starting at frame i: $[i, i + l^i - 1]$, and a similar window of successive frames of size l^j in the destination sequence ending at frame j: $[j - l^j + 1, j]$. These are the source and destination segments whose frames are to be blended gradually so as to generate a smooth transition between frames i and j. First, these two segment frames are temporally aligned with warps w^i and w^j respectively and with respect to the pose metric defined previously in section 3.1. We use a standard dynamic time warping algorithm [19] to estimate the best temporal warps. In addition, following [30] who argued that a transition duration in skeletal motion interpolation should be comprised between a third of a second and 2 seconds, we allow segment lengths l^i and l^j to vary between boundaries l_{min} and l_{max} that correspond to these values. We choose the transition with the least total surface deformation cost $D(i, j)$ through the interpolation path (see figure 2).

$$D(i,j) = \min_{l^i, l^j, w^i, w^j, L} \sum_{t \in [\![0, L]\!]} d(w^{i^{-1}}(t), w^{j^{-1}}(t)), \quad (4)$$

where $d(,)$ is the pose similarity metric defined in equation (3), and L is the transition duration, *i.e.* the length of the interpolation path found by the dynamic time warping optimization (see figure 2).

Since the transition duration L is uniquely defined for a given set of values (l^i, l^j, w^i, w^j), expression (4) rewrites:

$$D(i,j) = \min_{l^i, l^j \in [\![l_{min}, l_{max}]\!]} \underbrace{[\min_{w^i, w^j} \sum_{t \in [\![0, L]\!]} d(w^{i^{-1}}(t), w^{j^{-1}}(t)]}_{\text{Dynamic time warping}}.$$

The optimal parameters found with the above optimization, *i.e.* $(l^i_{i,j},\ l^j_{i,j},\ w^i_{i,j},\ w^j_{i,j},\ L_{i,j})$, will be used later on for motion synthesis.

For each pair of nodes i and j in the graph, we define the weight E_{ij} of the directed edge (i, j) as the weighted sum of the optimal surface deformation cost $D_{i,j} = D(i, j)$ and the associated duration cost $L_{i,j}$: $E_{ij} = D_{i,j} + \alpha L_{i,j}$. In the case where frames i and j belong to the same sequence, with $i < j$, it might be unnecessary to consider an interpolated transition if its cost is greater than $\alpha(j - i)$, that is the cost of going from i to j through the original motion sequence. The following expression summarizes the definition for the edge weight $E_{i,j}$ between poses i and j. If i and j do not belong to the same sequence :

$$\underbrace{E_{i,j}}_{\text{realism}} = \underbrace{D_{i,j}}_{\text{surface deformation}} + \alpha \underbrace{L_{i,j}}_{\text{duration}}, \quad (5)$$

Else :

$$E_{i,j} = \min[\alpha(j - i), D_{i,j} + \alpha L_{i,j}], \quad i < j. \quad (6)$$

The weight α represents the ratio of tolerance between surface deformation and transition duration. This user defined coefficient allows for some flexibility on the admissible surface deformation during a transition and with respect to time duration.

At this stage all nodes are connected in the graph with directed weighted edges, hence forming a complete digraph (see figure 1). The resulting structure is analogous to a quasi-semi-metric space, (I, E), where I is the set of nodes and $E : I \times I \to \mathbb{R}^+$ is the asymmetric function defined in equations(5) and(6). In the next stage, we proceed with the extraction of the essential subgraph from this complete digraph.

3.3. Essential subgraph

In a traditional motion graph, edges between sequences are obtained by selecting minima in transition cost matrices between pairs of sequences. This strategy is intuitive but yet not globally optimal, as transitions between sequences other than minima might give better total cost to a graph walk if the minimized criterion considers also duration for instance, which is an important constraint for motion synthesis. We propose therefore a global and principled strategy that consists of extracting the best paths between any pairs of poses and to keep only edges in the graph that contribute to these paths. This corresponds to extracting the essential sub-graph from the complete digraph induced from the input sequences. Unlike previous methods, ours ensures the

existence of at least one transition between any two nodes in the graph, which potentially yields to a better use of the original data with less dead ends trimming.

For every pair of nodes in the graph, we use Dijkstra algorithm to find the path with the least cost joining the source node to the destination node. The cost of a path $p = [n_1, n_2, ..., n_N]$ that goes through nodes $n_1, n_2, ..., n_N$ is defined as:

$$J(p) = J([n_1, n_2, ..., n_N]) = \sum_{i \in [\![1, N-1]\!]} E_{n_i, n_{i+1}}. \quad (7)$$

Once the optimal paths joining all pairs of nodes are found, all edges that do not belong to any of these paths are pruned. This gives an optimally connected graph which contains only the necessary edges to connect every pair of nodes with the best possible path cost in terms of realism (see figure 1). This resulting structure is also referred to as *the union of shortest path trees rooted at every graph node*, and its density is proportional to the α factor introduced in equations 5 and 6. The following sections explain how to use this structure to generate new animations.

4. High-level constraints

We consider in this part constrained navigation in the essential graph with various user-specified constraints such as spatial, temporal and behavioural constraints. We cast motion extraction as a graph search problem that consists of finding the walk $p = [n_1, n_2, ..., n_N]$ with minimal total error cost $J_c(p)$ as follows:

$$J_c(p) = J_c([n_1, n_2, ..., n_N]) = \sum_{i \in [\![1, N]\!]} j_c([n_1, ..., n_{i-1}], n_i), \quad (8)$$

where $j_c([n_1, ..., n_{i-1}], n_i)$ is a scalar function evaluating the additional error of adding node n_i to walk $[n_1, ..., n_{i-1}]$ with respect to the user specification. Since a graph node might be visited more than once in this search, a halting condition must be specified to prevent the search from infinite recursion. In order to demonstrate the interest of essential graphs for animation purposes, we implement two scenarios with different types of constraints. In both scenarios, the graph search was solved using Depth-First algorithm with Branch-and-Bound to reduce the computational complexity.

3D Behavioural path synthesis: In this scenario (see figure 5) the user provides a 3D curve that the character's center of mass should follow as closely as possible. Additionally, some parts of this path may impose specific types of motion taken from the dataset. This kind of constraints better adapts to datasets with a variety of basic motions involving locomotion activities, and for which there are significant displacements of the character in space. This is the case with our dataset *Thomas*. In this example, the cost function jc measures the 3D distance between the added frame to the current generated path and the point along the input 3D curve with the same arc length as the added frame along the generated path.

Pose/Time constraint: In this scenario the user provides a sequence of poses with associated times of occurrence. The search generates the motion sequence that satisfy as much as possible these pose/time constraints. This type of constraint adapts well to sequences of random unplanned activities, such as dance footages as in our dataset *Cathy*. The search problem is solved here for every two successive query poses separately, and the cost function j_c measures the duration cost of adding a node to the current generated path. The halting condition occurs when the duration of the travelled path between source and destination poses exceeds the desired duration.

5. Motion synthesis

In this section, we explain how a walk in the essential subgraph is converted into a continuous stream of motion in the form of a 3D mesh sequence. A walk path is represented by a node sequence, where every pair of successive nodes should correspond to an edge in the essential subgraph. We browse the path and sequentially generate the purely original or newly interpolated motion segments corresponding to every pair of consecutive nodes in the path. Interpolated segments are generated using the optimal transition scheme introduced in section 3.2. That is, temporally warped source and destination motion segments are blended by gradually aligning and interpolating the matched frame meshes as well as the displacement of their respective centres of mass. Finally, we perform a rigid alignment (as explained in section 3.1) at the junction between two consecutive motion segments to obtain a spatially continuous motion stream.

Pose interpolation: In order to interpolate frame meshes, we use a variant of Poisson shape interpolation [31]. Our experiments demonstrate that this strategy performs better compared to linear vertex interpolation or as rigid as possible approaches [24]. Using gradient field manipulation [3] notations, the 3×3 per-face gradient of the mesh for triangle m can be expressed as follow:

$$G_m = \begin{pmatrix} (v_1 - v_3)^T \\ (v_2 - v_3)^T \\ n^T \end{pmatrix}^{-1} \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} {v_1}^T \\ {v_2}^T \\ {v_3}^T \end{pmatrix}. \quad (9)$$

A global $3M \times N$ gradient operator G can then be yielded, where M and N are the total numbers of mesh triangles and vertices respectively:

$$\begin{pmatrix} G_1 \\ \vdots \\ G_M \end{pmatrix} = G \begin{pmatrix} v_1^T \\ \vdots \\ v_N^T \end{pmatrix}. \quad (10)$$

In order to interpolate poses with interpolation parameter $\lambda \in [0, 1]$, the interpolated rotation and scale/shear, extracted previously from per-face gradient deformations (see

section 3.1), are applied to each per-face gradient which yields new gradients $\tilde{G}_m$:

$$\tilde{T}_m = e^{\lambda \log(R_m)} \left((1-\lambda) I_3 + \lambda S_m\right), \quad (11)$$

$$\tilde{G}_m = \tilde{T}_m G_m. \quad (12)$$

The following standard Poisson system is then solved for new mesh vertex positions $\tilde{v}_1$ ... $\tilde{v}_1$, with a Dirichlet boundary condition imposing to the interpolated mesh center of mass to be at the same location as the original one.

$$\underbrace{G^T D G}_{\Delta} \begin{pmatrix} \tilde{v}_1^T \\ \vdots \\ \tilde{v}_N^T \end{pmatrix} = \underbrace{G^T D}_{div} \begin{pmatrix} \tilde{G}_1 \\ \vdots \\ \tilde{G}_M \end{pmatrix}, \quad (13)$$

where D is a diagonal $3M \times 3M$ mass matrix containing the triangle areas. We note that the left hand operator of this equation is equivalent to a discrete Laplace-Beltrami operator with cotangent weights. To insure the unicity of the solution, we perform a double interpolation using the source and target meshes and we linearly interpolate the resulting meshes.

6. Results

To evaluate our framework, we recorded two datasets of temporally consistent mesh sequences, with 5000 vertices and 10000 faces per mesh, and a frame rate of 50fps. These datasets were obtained using multi-view reconstruction and temporal tracking. The *Thomas* dataset contains 2633 frames of a male actor performing basic locomotion activities, and the *Cathy* dataset has 1910 frames of a female actress performing random dancing. We also applied our method to standard datasets of 3D mesh sequences, such as *Dan* and *JP* datasets [25] [4], the *Capoeira dancer* dataset [26] and the *Adobe dataset* [29].

Figure 3. (a) Our transition with Dynamic time warping and variable length blended segments, (b) Standard interpolated transition, both using Poisson pose interpolation

Transition approach: Our transition approach combining dynamic time warping and variable length blended segments offers substantial improvements in the visual results. Compared to a standard interpolated transition, our transitions show less visual flaws such as surface distortion (see figure 3) and foot-skate, and also compensates for the differential in velocity between source and destination motions through temporal warping.

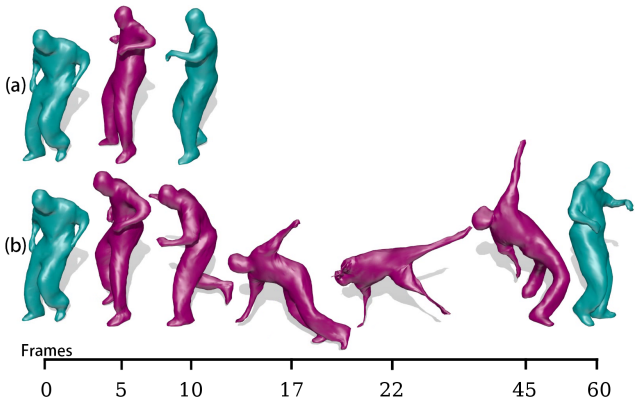

Figure 4. Illustration of motion graphs failures to find a graph walk as short and costless as the essential graph regardless of the transition approach. Input poses are in green. Note that input destination pose rotation around the vertical axis is estimated by the algorithm in both cases. (a) Essential graph, (b) Motion graph.

Graph structure: Given a dataset of mesh sequences, we want to generate motions joining all frames minimizing a joint cost of surface interpolation and duration. To this end, we construct an essential graph and a motion graph, and compute the costs of optimal paths joining every pair of frames in the dataset. In order to compare graph structures independently of the transition approach, we use the same transition cost for both methods. To construct a motion graph, we compute transition cost matrices between all pairs of sequences and select the local minima within them as transition points. In the original paper implementation, it is also recommended to only accept local minima below an empirically determined threshold to reduce the computational burden. We deliberately ignore this measure to obtain a highly connected motion graph to better challenge our method. Note that we do not compare to existing extensions of the motion graph since our primary objective is to evaluate the initial structure that is used to select optimal transitions and we believe that many of these extensions could anyway apply to the essential graph as well. Figure 6 presents the total costs of all the optimal paths using both our method and the standard motion graph. The plots show clearly that our method outperforms standard motion graphs quantitatively for various values of the α parameter that weights surface deformation and path duration contributions in the cost. Additionally, we show some visual results where motion graphs fail to find graph walks as short and costless as essential graphs (see figure 4).

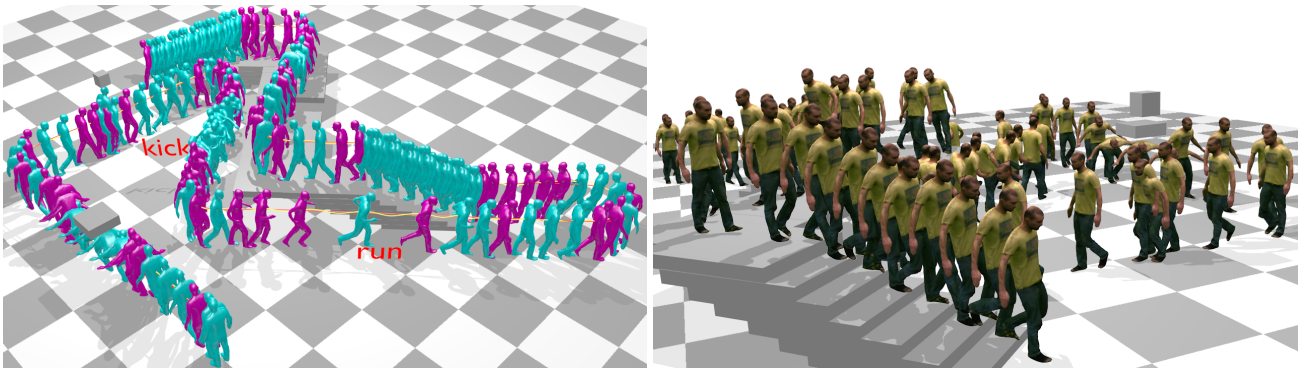

Figure 5. 3D Behavioural path synthesis (Section 4). Green meshes are original and red ones are interpolated.

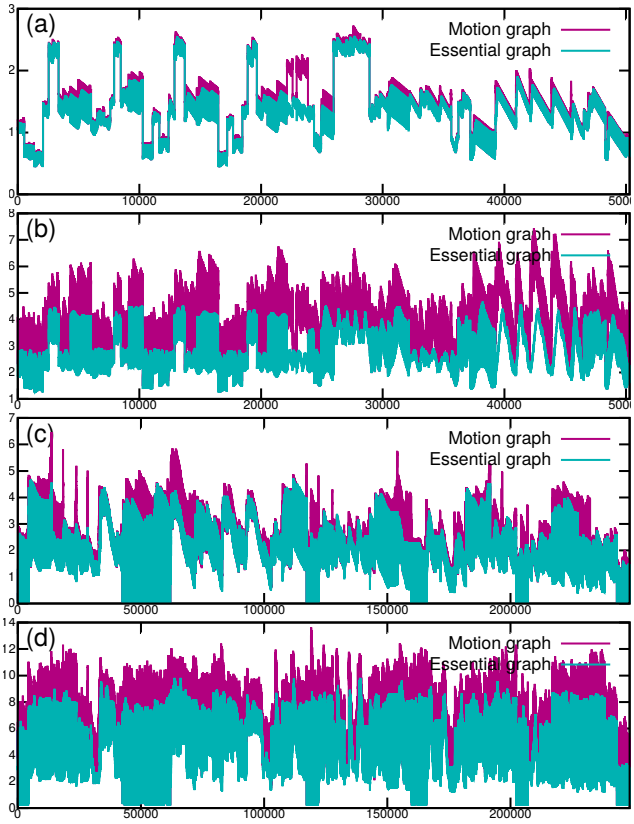

Figure 6. Comparisons between the essential graph a standard motion graph. Figures (a)-(d) show the costs (ordinate) of all optimal paths (abscissa) joining all pairs of frames in the dataset using using both methods while varying α. *Dan* dataset: (a) $\alpha = 0.01$, (b) $\alpha = 0.1$. *JP* dataset: (c) $\alpha = 0.01$, (d) $\alpha = 0.1$.

7. Conclusion

We presented a new strategy for automatic motion generation using temporally aligned 3D mesh sequences. This method introduces the essential graph as the organizing data structure for motion information. This structure identifies all the shortest paths in the fully connected graph of shape poses. It appears to be a good alternative to motion graphs that improves realism and better exploits the input information. In addition, the method proposes interpolated transitions between 3D surface meshes in place of frame concatenations.

Our contributions also include an approach to estimate frame to frame transition using Poisson pose interpolation. It combines variable-length source and destination motion segments along with dynamic time warping. In addition, we present a new motion transition cost function that evaluates a realism criterion. Although there is no standard objective way of measuring motion transition realism, neither for skeletal data nor for 3D surface data, other than perceptual evaluation, the criterion that we introduced in this work achieved satisfactory results in that respect and over the datasets we experimented. In practice, we have observed only very few mesh distortions and abrupt changes in the generated motions while such artifacts are often present with standard interpolated transitions.

Nonetheless, we have also experienced artifacts which might appear during transitions and such as foot-skate, which is a common issue in motion synthesis with multi-target blending. With skeletal data, this issue is partially dealt with by annotating the input data with contact information and taking this annotation into consideration while generating transitions, and also by applying inverse-kinematics on transitions to erase the foot-skate effect from them [16]. Although mesh sequence annotation seems to be more complicated, a similar approach could be applied on 3D mesh sequences.

References

[1] O. Arikan and D. A. Forsyth. Interactive motion generation from examples. *ACM Trans. Graph.*, 21(3), 2002. 3

[2] A. H. Barr. Global and local deformations of solid primitives. In *Proc. of ACM SIGGRAPH*, 1984. 4

[3] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1), 2008. 6

[4] C. Budd, P. Huang, M. Klaudiny, and A. Hilton. Global non-rigid alignment of surface sequences. *Int. J. Comput. Vision*, 102(1-3), 2013. 7

[5] D. Casas, M. Tejera, J. Guillemaut, and A. Hilton. Parametric control of captured mesh sequences for real-time animation. In *Motion in Games - 4th International Conference*, 2011. 4

[6] D. Casas, M. Tejera, J.-Y. Guillemaut, and A. Hilton. 4d parametric motion graphs for interactive animation. In *Proc. of the 2012 Symposium on Interactive 3D Graphics and Games*, 2012. 1, 4

[7] J. Gall, C. Stoll, E. De Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel. Motion capture using joint skeleton tracking and surface estimation. In *Proc. of CVPR*, 2009. 1

[8] M. Gleicher, H. J. Shin, L. Kovar, and A. Jepsen. Snap-together motion: Assembling run-time animations. *ACM Trans. Graph.*, 22(3), 2003. 4

[9] R. Heck and M. Gleicher. Parametric motion graphs. In *Proc. of the 2007 Symposium on Interactive 3D Graphics and Games*, 2007. 4

[10] P. Huang, A. Hilton, and J. Starck. Human motion synthesis from 3d video. In *Proc. of CVPR*, 2009. 1, 3, 4

[11] L. Ikemoto, O. Arikan, and D. Forsyth. Quick transitions with cached multi-way blends. In *Proc. of the 2007 Symposium on Interactive 3D Graphics and Games*, 2007. 3

[12] D. L. James, C. D. Twigg, A. Cove, and R. Y. Wang. Mesh ensemble motion graphs: Data-driven mesh animation with constraints. *ACM Trans. Graph.*, 26(4), 2007. 3

[13] L. Kovar and M. Gleicher. Flexible automatic motion blending with registration curves. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003. 3

[14] L. Kovar and M. Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph.*, 23(3), 2004. 3, 4

[15] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Trans. Graph.*, 21(3), 2002. 1, 2, 3

[16] L. Kovar, J. Schreiner, and M. Gleicher. Footskate cleanup for motion capture editing. In *Proc. of the 2002 ACM/Eurographics Symposium on Computer Animation*, 2002. 8

[17] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. *ACM Trans. Graph.*, 21(3), 2002. 3

[18] J. Min and J. Chai. Motion graphs++: A compact generative model for semantic motion analysis and synthesis. *ACM Trans. Graph.*, 31(6), 2012. 3

[19] M. Müller. *Information Retrieval for Music and Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. 5

[20] C. Ren, L. Zhao, and A. Safonova. Human motion synthesis with optimization-based graphs. *Comput. Graph. Forum*, 29(2), 2010. 3

[21] C. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen. Efficient generation of motion transitions using spacetime constraints. In *Proc. of ACM SIGGRAPH*, 1996. 3

[22] A. Safonova and J. K. Hodgins. Construction and optimal search of interpolated motion graphs. *ACM Trans. Graph.*, 26(3), 2007. 3

[23] H. J. Shin and H. S. Oh. Fat graphs: constructing an interactive character with continuous controls. In *Proc. of the 2006 ACM/Eurographics Symposium on Computer Animation*, 2006. 4

[24] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proc. of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 2004. 4, 6

[25] J. Starck and A. Hilton. Surface capture for performance-based animation. *IEEE Comput. Graph. Appl.*, 27(3), 2007. 7

[26] C. Stoll, J. Gall, E. de Aguiar, S. Thrun, and C. Theobalt. Video-based reconstruction of animatable human characters. *ACM Trans. Graph.*, 29(6), 2010. 1, 7

[27] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3), 2004. 4

[28] L. M. Tanco and A. Hilton. Realistic synthesis of novel human movements from a database of motion capture examples. In *Proc. of the Workshop on Human Motion (HUMO'00)*, 2000. 3

[29] D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph.*, 27(3), 2008. 1, 7

[30] J. Wang and B. Bodenheimer. Synthesis and evaluation of linear motion transitions. *ACM Trans. Graph.*, 27(1), 2008. 3, 5

[31] D. Xu, H. Zhang, Q. Wang, and H. Bao. Poisson shape interpolation. In *Proc. of the 2005 ACM Symposium on Solid and Physical Modeling*, 2005. 6

[32] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with poisson-based gradient field manipulation. In *Proc. of ACM SIGGRAPH*, 2004. 4

[33] L. Zhao and A. Safonova. Achieving good connectivity in motion graphs. In *Proc. of the 2008 ACM/Eurographics Symposium on Computer Animation*, July 2008. 3

[34] C. Zheng. One-to-many: example-based mesh animation synthesis. In *Proc. of the 2013 ACM/Eurographics Symposium on Computer Animation*, 2013. 3